Mumpy Documentation

Ian Ling

Mar 05, 2019

Contents

1 User Guide **1**
1.1 Installation 1
1.2 Usage 1
1.3 Examples 3
1.4 Development 4

2 API Reference **5**
2.1 Channel Object 5
2.2 MumpyEvent Enum 6
2.3 Mumpy Object 7
2.4 User Object 13

3 Indices and tables **15**

Python Module Index **17**

CHAPTER 1

User Guide

1.1 Installation

Mumpy has not been released on PyPI yet, so it must be cloned from the Git repo and installed manually.

```
$ git clone https://github.com/ianling/mumpy.git
$ cd mumpy/
$ python setup.py install
```

1.1.1 Requirements

- **Python 3.6+**
  - opuslib
  - pycryptodome
  - protobuf
- **libopus (for audio)**
  - Debian/Ubuntu: *apt install libopus0*
  - OSX: *brew install opus*
  - Windows: http://opus-codec.org/downloads/

1.2 Usage

Mumpy is a fully-featured Mumble client that offers both an API for interacting with a Mumble server, and an event-driven framework for reacting to various actions and situations as they occur.

The API portion contains all the features you would expect from a Mumble client, such as the ability to send and receive voice and text chat messages, kick/ban/mute/deafen users, create/edit/remove channels, and most everything else you can do in the official Mumble client.

```
from mumpy import Mumpy

my_mumpy = Mumpy()
my_mumpy.connect('localhost', port=64738)
my_mumpy.text_message("I am sending a text chat message to my current channel.")
my_mumpy.kick_user_by_name("BadUser1337", reason="Not good.")

# you can also interact with User and Channel objects in intuitive ways
bad_user = my_mumpy.get_user_by_name('BadUser1337')
bad_user.kick(ban=True)
my_channel = my_mumpy.get_current_channel()
my_channel.rename('New Channel Name')
```

A full list of all the methods available can be found in the *API Reference* section of the documentation.

The event-driven portion is essentially an alert system that allows you to run your own code in response to specific events happening. Some of these events include users connecting or disconnecting, people sending voice or text chat messages, people being kicked or banned, and new channels being created or removed.

A full list of all the events you can add handlers for can be found in the `MumpyEvent` part of the *API Reference* section.

Event handlers should always accept two parameters; the first parameter is the `Mumpy` instance that the event originated from, and the second is the protobuf message object that caused the event to fire. The fields you can expect to see in each protobuf message type are documented in the official Mumble client's protobuf definition file.

```
def kick_event_handler(mumpy_instance, raw_message):
    kicker = mumpy_instance.get_user_by_id(raw_message.actor)
    victim = mumpy_instance.get_user_by_id(raw_message.session)
    print(f"{kicker.name} kicked {victim.name} from the server!")

my_mumpy.add_event_handler(MumpyEvent.USER_KICKED, kick_event_handler)
```

Many parts of Mumpy operate asynchronously, so many of the functions do not return values themselves. For example, when you call the `update_user_stats()` method, a request for the user's stats is sent to the server. The server will eventually (usually within milliseconds) respond, which will trigger the `USER_STATS_UPDATED` event, where you can handle the values that the server sent back to us.

A (non-exhaustive) list of events you might see fired is included in each function's documentation in the *API Reference* section. If you would like a log all the events Mumpy is firing in real time, enable DEBUG logging output. See the *Logging* section below for more details.

1.2.1 SSL Certificates

Mumble allows clients to use an SSL certificate to verify their identity on the server. This also allows the server to remember which channel they were last in when they disconnected, and assign them various permissions on the server.

You can generate a self-signed SSL certificate and key file using a command like the following:

```
$ openssl req -newkey rsa:2048 -nodes -keyout mumpy_key.pem -x509 -days 2000 -out␣
↪mumpy_certificate.pem
```

To use the certificate and key file you generated, use the `certfile` and `keyfile` parameters when connecting to a server:

```
my_mumpy = Mumpy()
my_mumpy.connect('localhost', certfile='mumpy_certificate.pem', keyfile='mumpy_key.pem
↪')
```

1.2.2 Logging

Mumpy uses Python's logging library to handle logging. If you are seeing too many logs, you can add the following code to your program to reduce the logging verbosity:

```
import logging

logging.basicConfig(level=logging.WARNING)  # DEBUG, INFO, and ERROR are also valid
```

1.3 Examples

1.3.1 Barebones Connection

This example simply connects to a server, sends a text chat message to the channel, and then disconnects.

```
from mumpy import Mumpy

my_bot = Mumpy(username="MyBot")
my_bot.connect('localhost')  # port=64738 by default
my_bot.text_message("HELLO!")
my_bot.disconnect()
```

1.3.2 Barebones Connection Using 'with'

This example uses a different syntax to perform all the same actions as the example above.

```
from mumpy import Mumpy

with Mumpy() as my_bot:
    my_bot.connect('localhost')
    my_bot.text_message("Hello!")
```

1.3.3 Echo Bot

This example is a bot that echoes all text chat messages back to the original sender as a private message.

```
from mumpy import Mumpy, MumpyEvent
from time import sleep

def text_message_handler(mumpy_instance, raw_message):
    sender = mumpy_instance.get_user_by_id(raw_message.actor)
    message_body = raw_message.message
    mumpy_instance.text_message(message_body, users=(sender,))

my_bot = Mumpy(username="MyBot")
```

(continues on next page)

(continued from previous page)

```
my_bot.add_event_handler(MumpyEvent.MESSAGE_RECEIVED, text_message_handler)  # add␣
↪our function as a handler for MESSAGE_RECEIVED events
my_bot.connect('localhost')

while my_bot.is_alive():
    sleep(1)
```

1.3.4 Play WAV File

This example is a bot that connects to a server, waits for the UDP socket to become established, and then immediately transmits a WAV file. At the moment, WAV files must be in 48kHz 16-bit format.

```
from mumpy import Mumpy, MumpyEvent
from time import sleep

def udp_connected_handler(mumpy_instance, raw_message):
    mumpy_instance.play_wav('/home/ian/some_sound.wav')

my_bot = Mumpy(username="MyBot")
my_bot.add_event_handler(MumpyEvent.UDP_CONNECTED, udp_connected_handler)
my_bot.connect('localhost')

while my_bot.is_alive():
    sleep(1)
```

1.4 Development

Mumpy is open source under the GNU General Public License (GPL) version 3. The source code can be found on Github.

1.4.1 Contributing

If you have any contributions to make, whether they are bug reports, feature requests, or even code, feel free to submit issues and pull requests on Github.

This repo uses Travis CI to run a Python style checker called flake8. This checker looks for errors in the code, as well as deviations from the PEP8 style guide.

In order to style check your code locally before pushing it to Github, you can run a command like the following, from the root of the repo:

```
$ python3 -m flake8 .
```

We also ignore some of the flake8 style suggestions. Check the Travis config file in the repo to see exactly what flake8 command will get run on code pushed to the repo.

1.4.2 Building the Documentation

To build the documentation locally, enter the `docs/` directory and run the command `make html`.

This section contains information about installing and using Mumpy.

CHAPTER 2

API Reference

2.1 Channel Object

class mumpy.channel.**Channel**(*server*, *message=None*)

get_description()
Queries the server for the channel's description.

Returns None

get_users()
Retrieves a list of Users in this channel.

Returns a list of the Users in this channel

Return type list

id
This channel's ID.

remove()
Removes this channel from the server.

Returns None

rename(*new_name*)
Sets the channel's name to new_name.

Parameters **new_name** (*str*) – The new name for the channel

Returns None

2.2 MumpyEvent Enum

class mumpy.constants.**MumpyEvent**

The event types supported by Mumpy.

AUDIO_DISABLED = 'audio_disabled'
Fired when the client disables audio processing. This happens when the client fails to initialize the chosen audio codec, or does not support any of the server's audio codecs.

AUDIO_ENABLED = 'audio_enabled'
Fired when the client enables audio processing. This happens when the client initially connects to the server and successfully initializes an audio codec.

AUDIO_TRANSMISSION_RECEIVED = 'audio_transmission_received'
Fired when the client has received a complete audio transmission from the server.

AUDIO_TRANSMISSION_SENT = 'audio_transmission_sent'
Fired when the client has sent a complete audio transmission to the server.

BANLIST_MODIFIED = 'banlist_modified'
Fired when the server's ban list is modified.

CHANNEL_ADDED = 'channel_added'
Fired when a channel is added to the server.

CHANNEL_PERMISSIONS_UPDATED = 'channel_permissions_updated'
Fired when the Mumpy instance's permissions in a channel have changed.

CHANNEL_REMOVED = 'channel_removed'
Fired when a channel is removed from the server.

CHANNEL_UPDATED = 'channel_updated'
Fired when a channel is updated or modified in some way.

CONNECTED = 'self_connected'
Fired when the client has connected and authenticated successfully.

DISCONNECTED = 'self_disconnected'
Fired when the client has disconnected from the server. May be preceded by a USER_KICKED and a USER_BANNED event.

MESSAGE_RECEIVED = 'message_received'
Fired when a text message is received.

MESSAGE_SENT = 'message_sent'
Fired when the client sends a text message.

REGISTERED_USER_LIST_RECEIVED = 'registered_user_list_received'
Fired when the client receives the list of registered users on the server. These are stored in <Mumpy instance>.registered_users

UDP_CONNECTED = 'udp_connected'
Fired when the client has successfully established a UDP connection to the server

UDP_DISCONNECTED = 'udp_disconnected'
Fired when the client has lost or intentionally ended the UDP connection. This implies that audio communications have reverted back to using the TCP connection.

USER_AVATAR_UPDATED = 'user_avatar_updated'
Fired when a user changes their avatar.

USER_BANNED = 'user_banned'
Fired when anyone is banned from the server.

USER_COMMENT_UPDATED = 'user_comment_updated'
Fired when a user changes their comment.

USER_CONNECTED = 'user_connected'
Fired when someone else connects to the server.

USER_DEAFENED = 'user_deafened'
Fired when a user is deafened server side (e.g. by a server admin).

USER_DISCONNECTED = 'user_disconnected'
Fired when someone else disconnects from the server. May be preceded by a USER_KICKED and a USER_BANNED event.

USER_KICKED = 'user_kicked'
Fired when anyone is kicked from the server.

USER_MUTED = 'user_muted'
Fired when a user is muted server side (e.g. by a server admin).

USER_RECORDING = 'user_recording'
Fired when a user starts recording.

USER_REGISTERED = 'user_registered'
Fired when a user registers on the server.

USER_SELF_DEAFENED = 'user_self_deafened'
Fired when a user deafens themselves.

USER_SELF_MUTED = 'user_self_muted'
Fired when a user mutes themselves.

USER_SELF_UNDEAFENED = 'user_self_undeafened'
Fired when a user undeafens themselves.

USER_SELF_UNMUTED = 'user_self_unmuted'
Fired when a user unmutes themselves.

USER_STATS_UPDATED = 'user_stats_updated'
Fired when updated stats about a user are received. This happens after the client specifically requests stats about a user.

USER_STOPPED_RECORDING = 'user_stopped_recording'
Fired when a user stops recording.

USER_UNDEAFENED = 'user_undeafened'
Fired when a user is undeafened server side (e.g. by a server admin).

USER_UNMUTED = 'user_unmuted'
Fired when a user is unmuted server side (e.g. by a server admin).

USER_UNREGISTERED = 'user_unregistered'
Fired when a user is unregistered on the server.

2.3 Mumpy Object

class mumpy.mumpy.**Mumpy**(*username='mumble-bot'*, *password=''*)

add_event_handler(*event_type*, *function_handle*)

Adds the function as a handler for the specified event type. When an event is fired, any functions added as handlers for that event type will be run with two arguments, the Mumpy instance that the event originated from (in case you have multiple instances running), as well as the protobuf message that caused the event to be fired.

Example:

```
def kick_handler_function(mumpy_instance, raw_message):
    kicked_user = mumpy_instance.get_user_by_id(raw_message.session)
    kicker_session_id = raw_message.actor
    reason = raw_message.reason

bot.add_event_handler(MumpyEvent.USER_KICKED, kick_handler_function)
```

Parameters

- **event_type** (*str*) – an event from the *MumpyEvent* enum
- **function_handle** (*function*) – the function to run when the specified event is fired

Returns None

channel

the Channel the bot is currently in.

Type Returns

Type *Channel*

channel_id

the ID of the channel the bot is currently in

Type Returns

Type int

clear_all_audio_logs()

Clears every user's audio log, removing all received audio transmissions from memory.

connect(*address*, *port=64738*, *certfile=None*, *keyfile=None*, *keypassword=None*)

Starts the connection thread that connects to address:port. Optionally uses an SSL certificate in PEM format to identify the client.

Parameters

- **address** (*str*) – string containing either an IP address, FQDN, hostname, etc.
- **port** (*int*) – the TCP port that the server is running on (Default value = 64738)
- **certfile** (*str, optional*) – the path to the SSL certificate file in PEM format (Default value = None)
- **keyfile** (*str, optional*) – the path to the certificate's key file (Default value = None)
- **keypassword** (*str, optional*) – the secret key used to unlock the key file (Default value = None)

Returns None

deafen_self()

Deafens the Mumpy instance on the server.

Returns None

deafen_user(*user*)
Deafens a user on the server.

Parameters **user** (User) – the user to deafen

Returns None

disconnect()
Closes the connection to the server.

Returns None

export_audio_logs_to_wav(*folder='./'*)
Converts all audio logs from all users to WAV and saves them to separate files. Clears all audio logs once the audio has been saved.

Parameters **folder** (*str*) – the output directory (Default value = './')

Returns None

get_channel_by_id(*channel_id*)

Parameters **channel_id** (*int*) – the ID of the channel

Returns the Channel identified by channel_id

Return type *Channel*

get_channel_by_name(*name*)

Parameters **name** (*str*) – the name of the channel

Returns the Channel identified by name

Return type *Channel*

get_channel_permissions(*channel*)
Retrieves the Mumpy instance's permissions in the specified Channel. This function does not return anything. The server's response may fire the following events: - CHANNEL_PERMISSIONS_UPDATED

Parameters **channel** (Channel) – the Channel to retrieve permissions for

Returns None

get_channels()

Returns a dictionary of *Channel* objects and IDs in the form `<Mumpy>.get_channels()[id] = Channel()`

Return type dict

get_registered_users()
Retrieves the list of registered users from the server. This function does not return anything. The server's response may fire the following events: - REGISTERED_USER_LIST_RECEIVED

Returns None

get_user_by_id(*session_id*)

Parameters **session_id** (*int*) – the session ID of the user

Returns the User identified by session_id

Return type *User*

get_user_by_name(*name*)

Parameters **name** (*str*) – the name of the user

Returns the User identified by name

Return type *User*

get_users()

Returns a dictionary of `get_users()[id] = User()`

Return type dict

is_alive()

Returns True if bot is connected to the server

Return type bool

is_udp_alive()

Returns True if the bot has an active UDP connection to the server

Return type bool

join_channel(*channel*)

Moves the Mumpy instance to the specified Channel.

Parameters **channel** (`Channel`) – the channel to move to

Returns None

kick_user(*user*, *reason=''*, *ban=False*)

Kicks a User. Bans the User if ban is True.

Parameters

- **user** (`User`) – the target User
- **reason** (*str*) – the reason for this action (Default value = "")
- **ban** (*bool*) – whether or not the user should be banned as well (Default value = False)

Returns None

kick_user_by_name(*name*, *reason=''*, *ban=False*)

Kicks a user identified by name. Bans the user if ban is True.

Parameters

- **name** (*str*) – the target User's name
- **reason** (*str*) – the reason for this action (Default value = "")
- **ban** (*bool*) – whether or not the user should be banned as well (Default value = False)

Returns None

move_user_to_channel(*user*, *channel*)

Moves the User to the specified Channel.

Parameters

- **user** (`User`) – the User to move
- **channel** (`Channel`) – the channel to move the User to

Returns None

mute_self()

Mutes the Mumpy instance on the server.

Returns None

mute_user (*user*)
Mutes a user on the server.

Parameters **user** (User) – the user to mute

Returns None

ping (*udp=False*)
Sends a Ping packet to the server, as specified by the Mumble protocol.

Parameters **udp** (*boolean*) – if True, sends a UDP ping. Otherwise, sends a TCP ping. (Default value = False)

Returns None

play_wav (*filename*)
Reads a WAV file and then sends it as an audio transmission.

Parameters **filename** (*str*) – the path to the WAV file

Returns None

register_self ()
Registers the Mumpy instance on the server.

Returns None

register_user (*user*)
Registers a User on the server.

Parameters **user** (User) – the User to register

Returns None

remove_channel (*channel*)
Removes a channel.

Parameters **channel** (Channel) – the channel to remove

Returns None

rename_channel (*channel*, *new_name*)
Changes a channel's name to new_name.

Parameters

- **channel** (Channel) – the channel to rename
- **new_name** (*str*) – the new name

Returns None

request_blob (*user_textures=()*, *user_comments=()*, *channel_descriptions=()*)
Queries the server for the full contents of a User's texture or comment, or a Channel's description.

Parameters

- **user_textures** (*iterable*) – a list of Users to retrieve textures for (Default value = ())
- **user_comments** (*iterable*) – a list of Users to retrieve comments for (Default value = ())
- **channel_descriptions** (*iterable*) – a list of Channels to retrieve descriptions for (Default value = ())

Events

- USER_COMMENT_UPDATED
- USER_TEXTURE_UPDATED
- CHANNEL_UPDATED

send_audio(*pcm*, *sample_rate=48000*, *sample_width=2*)

Encodes raw PCM data using the preferred audio codec and transmits it to the server.

Parameters

- **pcm** (*bytes*) – the raw PCM data
- **sample_rate** (*int*) – the sample rate of the PCM data (Default value = 48000)
- **sample_width** (*int*) – the sample width of the PCM data (AKA the bit depth, but in bytes) (Default value = 2)

Returns None

text_message(*message*, *channels=()*, *users=()*)

Sends a text message to each Channel in the list channels, and to each User in the list users. If no channels or users are specified, sends the message to the bot's current channel.

Parameters

- **message** (*str*) – the text message
- **channels** (*iterable*) – a list of channels to send the message to (Default value = ())
- **users** (*iterable*) – a list of users to send the message to (Default value = ())

Returns None

undeafen_self()

Undeafens the Mumpy instance on the server.

Returns None

undeafen_user(*user*)

Undeafens a user on the server.

Parameters **user** (User) – the user to undeafen

Returns None

unmute_self()

Unmutes the Mumpy instance on the server.

Returns None

unmute_user(*user*)

Unmutes a user on the server.

Parameters **user** (User) – the user to unmute

Returns None

unregister_self()

Unregisters the Mumpy instance on the server.

Returns None

unregister_user(*user*)

Unregisters a User on the server.

Parameters **user** (User) – the User to unregister

Returns None

update_user_stats(*user*)
Queries the server for a User's stats. This function does not return anything. The server's response may fire the following events: - USER_STATS_UPDATED

Parameters **user** (User) – the User to retrieve stats for

Returns None

2.4 User Object

class mumpy.user.**User**(*server*, *message*)

channel
This user's current channel.

Returns the user's current channel

Return type *Channel*

clear_audio_log()
Clears this user's audio log, removing all their completed audio transmissions from memory.

Returns None

deafen()
Deafens the user.

Returns None

get_channel()
Get this user's current Channel.

Returns the user's current Channel

Return type *Channel*

kick(*reason=''*, *ban=False*)
Kicks user. Bans the user if ban is True.

Parameters

- **reason** (*str*) – The reason for kicking
- **ban** (*bool*) – Whether or not the user should also be banned

Returns None

move_to_channel(*channel*)
Moves the user to the specified channel.

Parameters **channel** (Channel) – the Channel to move them to

Returns None

mute()
Mutes the user.

Returns None

register()
Registers the user on the server.

Returns None

session_id
This user's session ID.

Returns session ID

Return type int

undeafen()
Undeafens the user.

Returns None

unmute()
Unmutes the user.

Returns None

update_comment()
Query the server for this user's comment.

Returns None

update_stats()
Requests updated stats about the user from the server.

Returns None

update_texture()
Query the server for this user's texture.

Returns None

This section contains information about the functions that Mumpy exposes to developers.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mumpy.channel, 5
mumpy.constants, 6
mumpy.mumpy, 7
mumpy.user, 13

Index

A

add_event_handler() (*mumpy.mumpy.Mumpy method*), 7
AUDIO_DISABLED (*mumpy.constants.MumpyEvent attribute*), 6
AUDIO_ENABLED (*mumpy.constants.MumpyEvent attribute*), 6
AUDIO_TRANSMISSION_RECEIVED (*mumpy.constants.MumpyEvent attribute*), 6
AUDIO_TRANSMISSION_SENT (*mumpy.constants.MumpyEvent attribute*), 6

B

BANLIST_MODIFIED (*mumpy.constants.MumpyEvent attribute*), 6

C

Channel (*class in mumpy.channel*), 5
channel (*mumpy.mumpy.Mumpy attribute*), 8
channel (*mumpy.user.User attribute*), 13
CHANNEL_ADDED (*mumpy.constants.MumpyEvent attribute*), 6
channel_id (*mumpy.mumpy.Mumpy attribute*), 8
CHANNEL_PERMISSIONS_UPDATED (*mumpy.constants.MumpyEvent attribute*), 6
CHANNEL_REMOVED (*mumpy.constants.MumpyEvent attribute*), 6
CHANNEL_UPDATED (*mumpy.constants.MumpyEvent attribute*), 6
clear_all_audio_logs() (*mumpy.mumpy.Mumpy method*), 8
clear_audio_log() (*mumpy.user.User method*), 13
connect() (*mumpy.mumpy.Mumpy method*), 8
CONNECTED (*mumpy.constants.MumpyEvent attribute*), 6

D

deafen() (*mumpy.user.User method*), 13
deafen_self() (*mumpy.mumpy.Mumpy method*), 8
deafen_user() (*mumpy.mumpy.Mumpy method*), 9
disconnect() (*mumpy.mumpy.Mumpy method*), 9
DISCONNECTED (*mumpy.constants.MumpyEvent attribute*), 6

E

export_audio_logs_to_wav() (*mumpy.mumpy.Mumpy method*), 9

G

get_channel() (*mumpy.user.User method*), 13
get_channel_by_id() (*mumpy.mumpy.Mumpy method*), 9
get_channel_by_name() (*mumpy.mumpy.Mumpy method*), 9
get_channel_permissions() (*mumpy.mumpy.Mumpy method*), 9
get_channels() (*mumpy.mumpy.Mumpy method*), 9
get_description() (*mumpy.channel.Channel method*), 5
get_registered_users() (*mumpy.mumpy.Mumpy method*), 9
get_user_by_id() (*mumpy.mumpy.Mumpy method*), 9
get_user_by_name() (*mumpy.mumpy.Mumpy method*), 9
get_users() (*mumpy.channel.Channel method*), 5
get_users() (*mumpy.mumpy.Mumpy method*), 10

I

id (*mumpy.channel.Channel attribute*), 5
is_alive() (*mumpy.mumpy.Mumpy method*), 10
is_udp_alive() (*mumpy.mumpy.Mumpy method*), 10

J

join_channel() (*mumpy.mumpy.Mumpy method*), 10

K

kick() (*mumpy.user.User method*), 13
kick_user() (*mumpy.mumpy.Mumpy method*), 10
kick_user_by_name() (*mumpy.mumpy.Mumpy method*), 10

M

MESSAGE_RECEIVED (*mumpy.constants.MumpyEvent attribute*), 6
MESSAGE_SENT (*mumpy.constants.MumpyEvent attribute*), 6
move_to_channel() (*mumpy.user.User method*), 13
move_user_to_channel() (*mumpy.mumpy.Mumpy method*), 10
Mumpy (*class in mumpy.mumpy*), 7
mumpy.channel (*module*), 5
mumpy.constants (*module*), 6
mumpy.mumpy (*module*), 7
mumpy.user (*module*), 13
MumpyEvent (*class in mumpy.constants*), 6
mute() (*mumpy.user.User method*), 13
mute_self() (*mumpy.mumpy.Mumpy method*), 10
mute_user() (*mumpy.mumpy.Mumpy method*), 11

P

ping() (*mumpy.mumpy.Mumpy method*), 11
play_wav() (*mumpy.mumpy.Mumpy method*), 11

R

register() (*mumpy.user.User method*), 13
register_self() (*mumpy.mumpy.Mumpy method*), 11
register_user() (*mumpy.mumpy.Mumpy method*), 11
REGISTERED_USER_LIST_RECEIVED (*mumpy.constants.MumpyEvent attribute*), 6
remove() (*mumpy.channel.Channel method*), 5
remove_channel() (*mumpy.mumpy.Mumpy method*), 11
rename() (*mumpy.channel.Channel method*), 5
rename_channel() (*mumpy.mumpy.Mumpy method*), 11
request_blob() (*mumpy.mumpy.Mumpy method*), 11

S

send_audio() (*mumpy.mumpy.Mumpy method*), 12
session_id (*mumpy.user.User attribute*), 14

T

text_message() (*mumpy.mumpy.Mumpy method*), 12

U

UDP_CONNECTED (*mumpy.constants.MumpyEvent attribute*), 6
UDP_DISCONNECTED (*mumpy.constants.MumpyEvent attribute*), 6
undeafen() (*mumpy.user.User method*), 14
undeafen_self() (*mumpy.mumpy.Mumpy method*), 12
undeafen_user() (*mumpy.mumpy.Mumpy method*), 12
unmute() (*mumpy.user.User method*), 14
unmute_self() (*mumpy.mumpy.Mumpy method*), 12
unmute_user() (*mumpy.mumpy.Mumpy method*), 12
unregister_self() (*mumpy.mumpy.Mumpy method*), 12
unregister_user() (*mumpy.mumpy.Mumpy method*), 12
update_comment() (*mumpy.user.User method*), 14
update_stats() (*mumpy.user.User method*), 14
update_texture() (*mumpy.user.User method*), 14
update_user_stats() (*mumpy.mumpy.Mumpy method*), 13
User (*class in mumpy.user*), 13
USER_AVATAR_UPDATED (*mumpy.constants.MumpyEvent attribute*), 6
USER_BANNED (*mumpy.constants.MumpyEvent attribute*), 6
USER_COMMENT_UPDATED (*mumpy.constants.MumpyEvent attribute*), 7
USER_CONNECTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_DEAFENED (*mumpy.constants.MumpyEvent attribute*), 7
USER_DISCONNECTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_KICKED (*mumpy.constants.MumpyEvent attribute*), 7
USER_MUTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_RECORDING (*mumpy.constants.MumpyEvent attribute*), 7
USER_REGISTERED (*mumpy.constants.MumpyEvent attribute*), 7
USER_SELF_DEAFENED (*mumpy.constants.MumpyEvent attribute*), 7

USER_SELF_MUTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_SELF_UNDEAFENED (*mumpy.constants.MumpyEvent attribute*), 7
USER_SELF_UNMUTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_STATS_UPDATED (*mumpy.constants.MumpyEvent attribute*), 7
USER_STOPPED_RECORDING (*mumpy.constants.MumpyEvent attribute*), 7
USER_UNDEAFENED (*mumpy.constants.MumpyEvent attribute*), 7
USER_UNMUTED (*mumpy.constants.MumpyEvent attribute*), 7
USER_UNREGISTERED (*mumpy.constants.MumpyEvent attribute*), 7